Fundamenta Informaticae 91 (2009) 145–159
DOI 10.3233/FI-2009-0037
IOS Press

Array P Systems and $t-$Communication

K. G. Subramanian[*], **Rosihan M. Ali**
School of Mathematical Sciences
Universiti Sains Malaysia, 11800 Penang, Malaysia
kgsmani1948@yahoo.com

Atulya K. Nagar
Deanery of Business and Computer Sciences,
Liverpool Hope University,
Hope Park, Liverpool L16 9JD, UK

Maurice Margenstern
LITA, Universite Paul Verlaine-Metz,
Ile du Saulcy, 57045 Metz Cedex, France

Abstract. The two areas of grammar systems and P systems, which have provided interesting computational models in the study of formal string language theory have been in the recent past effectively linked in [4] by incorporating into P systems, a communication mode called $t-$mode of cooperating distributed grammar systems. On the other hand cooperating array grammar systems [5] and array P systems [1] have been developed in the context of two-dimensional picture description. In this paper, motivated by the study of [4], these two systems are studied by linking them through the $t-$communication mode, thus bringing out the picture description power of these systems.

Keywords: Grammar systems, P systems, Picture languages, Array grammars

[*]Address for correspondence: School of Mathematical Sciences,Universiti Sains Malaysia, 11800 Penang, Malaysia and Deanery of Business and Computer Sciences, Liverpool Hope University, Liverpool L16 9JD UK.

1. Introduction

The theory of grammar systems [3] has provided an effective grammatical framework for capturing several phenomena characteristic of multi-agent systems. Different kinds of grammar systems have been introduced and investigated in the literature to capture various features such as cooperation, distribution, communication, parallelism and so on. On the other hand the new computability model introduced by Păun [11] inspired by the structure and functioning of living cells has proved to be a rich frame work for obtaining universality results and studying many computational problems. Although these two areas originated from different motivations and the models were formulated with different basic structures, recently these two have been linked in [4], by incorporating into rewriting P systems [11], a communication mode called $t-$mode of cooperating distributed (CD) grammar systems [3]. The power of the resulting P systems is investigated in [4] by comparing them with CD grammar systems.

In theoretical studies on generation and analysis of images or pictures in the two-dimensional plane, syntactic techniques have constituted one of the main areas of study. Motivated by different applications such as character recognition, pictorial information system design, pattern recognition and so on, several types of two-dimensional picture generating systems have been proposed in the literature [13, 14, 7, 2, 25, 24] extending to two-dimensions the well-known Chomskian string grammars [16] or Lindenmayer systems [15] or Marcus contextual grammars [6, 10] and others.

The power of the mechanism of cooperation in generating pictures by array grammars is investigated in [5]. It is shown in [5] that the power of cooperating array grammar systems is strictly greater than that of context-free array grammars. In fact the generative power is increased even in the case of systems with components having regular array grammar rules which is in contrast to the string case. On the other hand in [1], array P systems are considered extending the string-objects P systems to array-objects P systems, thus bringing together the two areas of membrane computing and picture grammars considered in the form of two-dimensional $(2d)$ array grammars. It is shown in [1] that P systems with array context-free rules are computationally universal. The array P system in [1] uses an extended alphabet which includes a terminal alphabet, has internal output with the result being obtained in a specified membrane and considers halting computations.

Motivated by the study in [4], here we incorporate into array P systems the $t-$mode of communication of cooperating array grammar systems, and examine the power of the array P systems, thus providing a natural extension of the study in [4] to picture arrays.

2. Basic Definitions and examples

We refer to [16] for notions of formal language theory and to [5] for array grammars and array languages. For notions pertaining to P systems, we refer to [11] and for grammar systems to [3].

An alphabet V is a finite set of symbols. A word or a string w over V is a sequence of symbols from V. The set of all words, including the empty word λ with no symbols, is denoted by V^* and $V^+ = V^* - \lambda$.

The pictures that we consider are arrays consisting of finitely many symbols from a given alphabet V with the symbols placed in the points of the plane and the points not marked with elements of V are assumed to have the *blank symbol* $\# \notin V$. An array is described by specifying the *pixels* v of nonblank points, together with their associated symbols from V. We will also use the intuitive pictorial

representation for 2D finite arrays indicating their non-blank pixels. Also we take into account only the relative positions of non-blank pixels in the array. For example, the T-shaped array with equal arms in Figure 1 is formally given by

$$\{((0,4),a),\ ((1,4),a),\ (2,4),a),\ ((3,4),a),\ ((4,4),a),\ ((5,4),a),\ ((6,4),a),$$

$$((7,4),a),\ ((8,4),a),\ ((4,0),a),\ ((4,1),a),\ ((4,2),a),\ ((4,3),a)\}$$

$$\begin{array}{ccccccccc} a & a & a & a & a & a & a & a & a \\ & & & & a & & & & \\ & & & & a & & & & \\ & & & & a & & & & \\ & & & & a & & & & \end{array}$$

Figure 1: T-shaped array with equal arms

We denote by V^{+2} the set of all two-dimensional non-empty finite arrays over V. The empty array is denoted by λ, and then the set of all arrays over V is $V^{*2} = V^{+2} \cup \{\lambda\}$. Any subset of V^{*2} is called an *array language*.

The array grammars we consider here are extensions of string grammars to two dimensional pictures [5, 2, 26].

An *array rewriting rule* p over V is of the form $p : \mathcal{A} \longrightarrow \mathcal{B}$ where $\mathcal{A}$ and $\mathcal{B}$ are arrays over V. For two arrays $\mathcal{C}, \mathcal{D}$ over V and a rule p as above, we write $\mathcal{C} \Longrightarrow_p \mathcal{D}$ ($\mathcal{C} \Longrightarrow \mathcal{D}$ when p is understood), if $\mathcal{D}$ can be obtained by replacing a subarray of $\mathcal{C}$ identical to $\mathcal{A}$ with $\mathcal{B}$. The reflexive and transitive closure of the relation $\Longrightarrow$ is denoted by $\Longrightarrow^*$.

An array production $p = (W, \mathcal{A}, \mathcal{B})$ is called:

1. *monotonic*, if the symbol positions of $\mathcal{A}$ are all contained in $\mathcal{B}$;
2. *$\#$-context-free*, if there is exactly one nonblank symbol in $\mathcal{A}$;
3. *context-free*, if it is both monotonic and $\#$-context-free.

An array grammar is a five-tuple $G = (N, T, \#, \{((0,0), S)\}, P)$, where N, T are disjoint alphabets of nonterminal symbols and of terminal symbols, respectively, $\# \notin N \cup T$ is the blank symbol, $S \in N$, and P is a finite set of array productions $\mathcal{A} \longrightarrow \mathcal{B}$ such that at least one pixel of $\mathcal{A}$ is marked with an element of N; usually, the *axiom array* $\{((0,0), S)\}$ will be simply written as S.

An array grammar is monotonic, $\#$-context-free, or context-free if all its rules are of these types; clearly, in the case of $\#$-context-free and context-free grammars, there is a unique non-blank pixel in the left hand array of each rule, marked with a nonterminal. *Regular* array grammar rules are of the following forms: $A\ \# \rightarrow a\ B,\ \#\ A \rightarrow B\ a,\ \begin{array}{c} \# \\ A \end{array} \rightarrow \begin{array}{c} B \\ a \end{array},\ \begin{array}{c} A \\ \# \end{array} \rightarrow \begin{array}{c} a \\ B \end{array},\ A \rightarrow B,\ A \rightarrow a$, where A, B are nonterminals and a is a terminal.

The array language generated by G is

$$L(G) = \{\mathcal{A} \in T^{*2} \mid \{((0,0), S)\} \Longrightarrow^* \mathcal{A}\}.$$

The families of array languages generated by arbitrary, monotonic, #-context-free, context-free, and regular array grammars are denoted by $ARE, AMON, A\#CF, ACF, AREG$ respectively. The following strict inclusions are known: $AREG \subset ACF \subset AMON \subset ARE$, $ACF \subset A\#CF \subset ARE$. Here we deal with only the families ACF and $AREG$.

The two-dimensional matrix grammars introduced in [17] and studied extensively in the literature (to quote a few [18, 8, 20, 19]) generate only rectangular arrays over terminals. We briefly recall these grammars. A $2d$ matrix grammar has two components (G_1, G_2) where G_1 is a Regular, CF or CS grammar generating in the first phase strings over a set of symbols $\{S_1, S_2, \cdots, S_k\}$, called intermediates; $G_2 = (G_{21}, G_{22}, \cdots, G_{2k})$ where each G_{2i} is a regular grammar corresponding to S_i and has right linear production rules of the form $X \rightarrow aY$ or $X \rightarrow a$ where X, Y are nonterminals of G_{2i} and a is a terminal of G_{2i}.

G is a regular, context-free, context-sensitive $2d$ matrix grammar if G_1 is regular, context-free, context sensitive respectively. Derivations are defined as follows: First a string over the intermediates of G_1 is generated horizontally using the rules of G_1. Vertical derivations then proceed in parallel using the rules of G_{2i} generating rectangular arrays over terminal symbols of $G_{2i}s$ when the vertical derivation terminates.

The set $L(G)$ consists of all $m \times n$ arrays generated by G. We denote the picture language classes of regular, CF $2d$ Matrix grammars by $2dRML$, $2dCFML$, respectively.

Now we very briefly mention the notion of cooperating array grammar system defined in [5]. The idea is analogous to string cooperating grammar system [3] except that array rewriting rules are taken in the components of the system. The family of array languages generated by cooperating array grammar systems consisting of at most n components with the rules in the components either regular or CF array rewriting rules and in the $t-$mode of derivation, is denoted by $CD_n(REGA, t)$ and $CD_n(CFA, t)$ according as all the components use regular rules or at least one component uses non-regular CF rules. In [5] an extensive investigation of the power of these systems is done.

We now briefly and informally recall a specific basic model of string-rewriting P systems and computations carried out in such systems. A comprehensive account of these systems can be found in [11] or at the web address `http://psystems.disco.unimib.it`.

In the *regions* of a string-rewriting P system defined by a hierarchical arrangement of *membranes*, *objects* which are strings of symbols are processed by *rewriting rules* (or other string handling operations) associated with the membranes. A typical rewriting rule used in a string-objects membrane system (also called *P system*) is of the form $X \rightarrow u(tar)$, where $X \rightarrow u$ is a usual context-free rule, and $tar \in \{here, out, in\}$ is a target indication, specifying the region where the result of the rewriting should be placed in the next step: *here* says that the result remains in the same compartment where the rule was applied, *out* says that the string has to be sent to the region surrounding the region where it has been produced, and *in* says that the string should go to one of the directly inner membranes, if any exists. By a command *out* a string can leave the system which happens when it is produced in the external membrane of a system. Each string is processed by at most one rule at a time; if any rule can be used, then one of these, nondeterministically chosen, is used; if no rule is available to rewrite a string, then it remains

unchanged. All strings, from all regions, are rewritten at the same time. A sequence of such steps is called a computation; a computation provides a result only if it halts, and a configuration of the system is reached where no further rule can be applied. The result of a computation is the set of strings present in the halting configuration in a specified elementary membrane; if acceptance is defined by collecting only the strings over a specified terminal alphabet, then the system is called extended and otherwise, it is called non-extended. We have described here a specific type of rewriting P systems, namely one with *internal output* in which the result is obtained in a specified membrane and with *halting* computations and this model has proved to be very useful in the array case.

We now very briefly mention computation in the basic model of a rewriting array-objects P system introduced in [1].

A computation in an array P system Π is defined in the same way as in the string rewriting P system described earlier with the successful computations being the halting ones: In the compartments of a membrane structure we place arrays, which evolve by means of array rewriting rules. The result of a computation is the set of arrays collected in a specified elementary membrane in the halting configuration.

The family of all array languages generated by systems Π as above, with at most m membranes, with rules of type $\alpha \in \{REG, CF\}$ is denoted by $EAP_m(\alpha)$ where REG, CF respectively refer to regular and context-free array rewriting rules ; if non-extended systems are considered (that is, we have $V = T$; in such a case we ignore the condition to have at least one nonterminal pixel in the left hand side of rules), then we write $AP_m(\alpha)$.

3. Array P system with t mode of communication

Csuhaj-Varju et al [4] have brought out the relationsihp between coperating distributed string grammar systems and string-objects P systems by linking them through the $t-$derivation mode. Dassow et al [5] have incorporated the $t-$mode of derivation in their study of cooperating array grammar systems and have demonstrated the power of this mechanism of $t-$mode of derivation, besides establishing other results. Motivated by the studies in [4], we now introduce an array P system with t mode of communication.

A P system (of degree $m \geq 1$) with array-objects and $t-$communication (or in short, a $t-$communicating array P system of type α, $(tEAPS_m(\alpha, \beta)), \alpha \in \{tin, tout\}, \beta \in \{REG, CF\}$ is a construct

$$\Pi = (V, T, \#, \mu, F_1, \ldots, F_m, R_1, \ldots, R_m, i_o),$$

where:

1. V is the total alphabet,
2. $T \subseteq V$ is the terminal alphabet,
3. $\#$ is the blank symbol,
4. μ is a membrane structure with m membranes labelled in a one-to-one way with $1, 2, \ldots, m$,
5. $F_1, \ldots, F_m$ are finite sets of arrays over V associated with the m regions of μ,

6. $R_1, \ldots, R_m$ are finite sets of array rewriting rules over V associated with the m regions of μ; the array-rewriting rules are context-free of the form $\mathcal{A} \rightarrow \mathcal{B}$ or $\mathcal{A} \rightarrow \mathcal{B}(tar)$ and $tar \in \{out, in\}$; In a given system at most one of the target indications in and out may be present in the rules of the regions.
7. i_o is the label of an elementary membrane of μ (the output membrane).

The computation starts in the usual way with the arrays, if any, of F_i initially present in the ith region $1 \leq i \leq m$. Each array in every membrane region i is rewritten by any available array-rewriting rule in the region R_i; only one rule, nondeterministically chosen, is applied to each array in every region. We now describe the manner in which the arrays are communicated or sent among the regions.If an array-rewriting rule has target indication in, then the array to which it is applied is non-deterministically sent to one of the immediately direct inner regions and if no such region exists such a rule cannot be applied. If an array-rewriting rule has target indication out, then the array to which it is applied is sent to its immediately direct upper region. If an array-rewriting rule has no target indication, then the array to which it is applied remains in the same region if it can be further rewritten there but if no rule can be applied to it in that region, then one of the following actions is done depending on the system is of type tin or $tout$.

1. A $t-$communicating array P system of type tin has array-rewriting rules in its regions with target indication out or no target indication (and does not have rules with target indication in). In fact when an array cannot be further rewritten in a region, it is sent to the immediately direct inner region if one such region exists. In other words the $t-$mode or maximal derivation performed enforces the in target command. If the membrane is elementary, the rewritten array remains there.
2. A $t-$communicating array P system of type $tout$ has array-rewriting rules in its regions with target indication in or no target indication (and does not have rules with target indication out). In fact when an array cannot be further rewritten in a region, it is sent to the immediately direct outer region if one such region exists. In other words the $t-$mode or maximal derivation performed enforces the out target command.

Remark 3.1. In defining correct computations, the halting condition has proved to be useful in the array case as has been noted in [1]. Here we therefore deal only with $t-$communicating array P system of type tin in the rest of the study and so we write $tEAPS_m(REG), tEAPS_m(CF)$, with the understanding that the type of the system is tin.

The set of all arrays computed or generated by a $t-$communicating P system Π is denoted by $tAL(\Pi)$. The family of all array languages $tAL(\Pi)$ generated by such systems Π with at most m membranes and rules of type $\alpha \in \{REG, CF\}$ is denoted by $tEAP_m(\alpha)$.

Example 1. Consider the $t-$communicating array P system, $tEAPS_4(REG)$,

$$\Pi_1 = (\{A, B, C, A', B', C', X, Y, a\}, \{a\}, \#, [_1[_2[_3[_4]_4]_3]_2]_1, \left\{ \begin{array}{c} A\,X\,B \\ C \end{array} \right\}, \emptyset, \emptyset, \emptyset, R_1, R_2, R_3, R_4, 4),$$

$$R_1 = \{\ \#\,A \ \rightarrow\ A'\,a\ ,\ B\,\# \ \rightarrow\ a\,B'\ ,\ \begin{array}{c} C \\ \# \end{array} \rightarrow \begin{array}{c} a \\ C' \end{array}\ \}$$

$R_2 = \{\ A' \ \rightarrow \ A\ ,\ B' \ \rightarrow \ B\ ,\ C' \ \rightarrow \ C\ ,\ Y \ \rightarrow \ X\ (out)\}$
$R_3 = \{\ X \ \rightarrow \ Y\ (out),\ X \ \rightarrow \ a\ \}$
$R_4 = \{\ A \ \rightarrow \ a\ ,\ B \ \rightarrow \ a\ ,\ C \ \rightarrow \ a\ \}$

Starting with the array $\left\{ \begin{array}{c} A\,X\,B \\ C \end{array} \right\}$ initially present in region 1 of the system, the "horizontal" and the "vertical" "arms" are grown once and due to the maximal mode of derivation and type tin of the system, the array is sent to the inner region 2 wherein the cells with labels A', B', C' are respectively changed into A, B, C. Again due to the tin type the array is sent to region 3. If the rule used is $X \ \rightarrow \ Y$, the target indication out sends it immediately to region 2 wherein the rule that can be used is only $Y \ \rightarrow \ X$ with target indication out and so the array is sent to region 1. The process repeats. But if the rule used in region 3 is $X \ \rightarrow \ a$, then the array enters region 4, the nonterminals are changed into the terminal a and the array generated is in the form of the letter T (Figure 1) with all three "arms" of equal length. These arrays constitute the language generated.

Example 2. Consider the $t-$communicating array P system, $tEAPS_4(CF)$,

$$\Pi_2 = (\{A, B, C, A', B', C', X, Y, a\}, \{a\}, \#, [_1[_2[_3[_4]_4]_3]_2]_1, \left\{ \begin{array}{c} A\,C \\ X\,B \end{array} \right\}, \emptyset, \emptyset, \emptyset, R_1, R_2, R_3, R_4, 4),$$

$$R_1 = \{\ \begin{array}{c} \# \\ A \end{array} \ \rightarrow \ \begin{array}{c} A' \\ a \end{array}\ ,\ \begin{array}{c} \#\,\# \\ C\,\# \end{array} \ \rightarrow \ \begin{array}{c} A'\,C' \\ a\,B' \end{array}\ ,\ B\ \#\rightarrow a\ B'\}$$

The rules in R_2, R_3, R_4 are as in example 1.

Starting with the array $\left\{ \begin{array}{c} A\,C \\ X\,B \end{array} \right\}$ initially present in region 1 of the system, the border made of the top row and rightmost column is grown just to increase the size of the array by one in both row and column, thus maintaining the "square" shape. The rest of the computation is similar to the example 1, finally yielding solid "square" arrays (Figure 2) over the terminal symbol a and these are collected in the region 4.

$$\begin{array}{cccccc} a & a & a & a & a & a \\ a & a & a & a & a & a \\ a & a & a & a & a & a \\ a & a & a & a & a & a \\ a & a & a & a & a & a \\ a & a & a & a & a & a \end{array}$$

Figure 2: A Solid square of $a's$

Remark 3.2. It is known that the set S_s of all $n \times n$ $(n \geq 2)$ solid squares over a, can be generated [26] by a regular array grammar. Here the $t-$communicating P system Π_2 of type tin also generates it. But

the difference is that the number of rules required in the former case is very large whereas in the case of Π_2, we have only 12 rules in all the 4 membranes together. It remains to be seen whether the number of rules can still be reduced.

The following results in Theorem 3.1 are immediate from the definition of $t-$communicating array P system.

Theorem 3.1. 1. For $X \in \{AREG, ACF\}$, $X \subseteq tEAP_1(X)$

2. For $Y \in \{REG, CF\}$, $tEAP_n(Y) \subseteq tEAP_{n+1}(Y), n \geq 1$

3. $tEAP_n(REG) \subseteq tEAP_n(CF), n \geq 1$

Theorem 3.2. 1. $2dRML \subset tEAP_4(CF)$

2. $tEAP_4(CF) - 2dCFML \neq \emptyset$

Proof:
Every $2dRML$ generated by a $2dRMG$ G can be simulated by a $tEAPS_4(CF)$ Π (of type tin) which is described below: We can assume that the rules in the first phase of the $2dRMG$ G are of the form $X \rightarrow AY$ or $X \rightarrow A$ where X, Y are nonterminals in the first phase of G and A is a terminal symbol (also called intermediate) of the first phase of G. Π has four membranes having a structure $[_1[_2[_3[_3[_4]_4]_3]_2]_1$. In regions 2, 3 and 4 initially there are no array-objects.

In the region 1 of Π, the start symbol of the $2dRMG$ is the initial array object. For every rule of the first phase of the $2dRMG$ of the form $X \rightarrow AY$, where X, Y are nonterminals of the first phase and A an intermediate, we include a rule $X\# \rightarrow AY$ in region 1. For each rule of the form $X \rightarrow A$, where X is a nonterminal of the first phase and A an intermediate, we include in region 1, a rule of the form $X \rightarrow [A, 1]$ where $[A, 1]$ is a new symbol and corresponds to A. Also rules of the of the forms $\begin{matrix} A \\ \# \end{matrix} \rightarrow \begin{matrix} a \\ B' \end{matrix}$, $\begin{matrix} [A,1] \\ \# \end{matrix} \rightarrow \begin{matrix} a \\ [B,1] \end{matrix}$ are included in region 1, corresponding to a regular rule $A \rightarrow aB$ in the second phase of the $2dRMG$, where $[A, 1], [B, 1]$ are new symbols that respectively correspond to A, B.

Region 2 has rules of the form $A' \rightarrow A$ where A' is a new symbol corresponding to the nonterminal A of the second phase of the $2dRMG$. Also rules of the forms $[A, 1] \rightarrow [A, 2], [A, 3] \rightarrow [A, 1](out)$ are included in region 2, for every such $[A, 1]$, where $[A, 2], [A, 3]$ are new symbols that correspond to $[A, 1]$.

In region 3, for every such symbol $[A, 2]$ created, a rule of the form $[A, 2] \rightarrow [A, 3](out)$ is added. Also, if $A \rightarrow a$ is a terminal rule in phase 2 of the $2dRMG$, we add a rule $[A, 2] \rightarrow a$ in region 3.

Region 4 is the output membrane. Region 4 contains all the terminal rules of the second phase of the $2dRMG$ G. The terminal symbols of Π are the terminal symbols of the $2dRMG$ and the nonterminals are the nonterminals, the intermediates of the $2dRMG$ and the newly introduced symbols.

It can be seen that due to the $t-$ mode of rewriting in the regions and the type tin of the system Π, the first phase of the $2dRMG$ is simulated in region 1 starting from start symbol (array) S except that in the strings generated the last symbol will be of the form $[A,1]$ for some intermediate A. Also one step of "vertical" derivation of the second phase of the $2dRMG$ is also simulated in this region. The resulting array is passed on to the region 2 wherein the nonterminals (of the second phase of $2dRMG$) are changed into their primed versions but the symbol of the form $[A,1]$ is changed into $[A,2]$. The result is passed on to region 3 wherein if the rule $[A,2] \rightarrow [A,3](out)$ is used, the array is sent back to region 2 and an application of the only possible rule $[A,3] \rightarrow [A,1](out)$ sends it to region 1 and the process is repeated. Otherwise, an application of the rule of the form $[A,2] \rightarrow a$ sends it to region 4 wherein the terminal rules of the second phase of the $2dRMG$ are used yielding the terminal arrays generated by the $2dRMG$. This shows the inclusion in the first statement.

The proper inclusion in the first statement is due to the fact that no $2dCFMG$ and hence no $2dRMG$ can generate the set S_s of solid squares over $\{a\}$ as the derivations in the two phases of the $2dRMG$ are independent of each other. But from example 2, $S_s \in tEAP_4(CF)$. This also proves the second statement. □

Remark 3.3. In [8], a variation of $2dRMG$ is considered by allowing "vertical" derivations to take place with the columns "growing" either up or down in parallel. This model can also be generated by a $tEAPS_4(CF)$ as in Theorem 3.2. The rules in the regions are to be suitably modified to take care of 'up' or 'down' rewriting in the variation [8] of the $2dRMG$.

Let R_h be the set of all hollow rectangles over the symbol $\{a\}$, a member of which is shown in Figure 3. Let S_h be the set of all hollow squares over the symbol $\{a\}$, in which all the sides are of equal length.

$$\begin{array}{cccccc} a & a & a & a & a & a \\ a & & & & & a \\ a & & & & & a \\ a & & & & & a \\ a & & & & & a \\ a & & & & & a \\ a & & & & & a \\ a & a & a & a & a & a \end{array}$$

Figure 3: A hollow rectangle of $a's$

Theorem 3.3. 1. $R_h \in tEAP_2(REG) \cap CD_2(REG,t)$

2. $tEAP_2(REG) - ACF \neq \emptyset$

Proof:
The set R_h is generated by the following $t-$communicating array P system $tEAPS_2(REG)$
$\Pi_3 = (\{S,A,X,B,Y,C,Z,D,F\}, \{a\}, [_1[_2]_2]_1, \{S\}, \emptyset, \emptyset, R_1, R_2, 2)$

$$R_1 = \{ \begin{matrix}\# \\ S\end{matrix} \rightarrow \begin{matrix}S \\ a\end{matrix}, \begin{matrix}\# \\ S\end{matrix} \rightarrow \begin{matrix}X \\ a\end{matrix}, X\# \rightarrow a\,A, A\# \rightarrow a\,A,$$
$$A\# \rightarrow a\,Y, \begin{matrix}Y \\ \# \end{matrix} \rightarrow \begin{matrix}a \\ B\end{matrix}, \begin{matrix}B \\ \# \end{matrix} \rightarrow \begin{matrix}a \\ B\end{matrix}, \begin{matrix}B \\ \# \end{matrix} \rightarrow \begin{matrix}a \\ Z\end{matrix},$$
$$\#Z \rightarrow C\,a, \#C \rightarrow C\,a, \#C \rightarrow D\,a, \begin{matrix}\# \\ D\end{matrix} \rightarrow \begin{matrix}F \\ a\end{matrix} \}$$
$$R_2 = \{D \rightarrow a\}$$

Starting from S and using rules of region R_1 a vertical line of $a's$ (the leftmost column of a rectangle) is grown until the generation turns right on an application of the rule $X\# \rightarrow a\,A$ to grow a horizontal line of $a's$ (the topmost row of the rectangle). The application of the rule $\begin{matrix}Y \\ \# \end{matrix} \rightarrow \begin{matrix}a \\ B\end{matrix}$ turns the generation down to grow again a vertical line of $a's$ (the rightmost column of the rectangle) until the application of the rule $\#Z \rightarrow C\,a$ which turns the generation to the left again to grow a horizontal line of $a's$ (the bottommost row of the rectangle). If the generation ends in region 1 with a correct application of the rule $\#C \rightarrow D\,a$ (the correct application is indicated by non-applicability of the rule $\begin{matrix}\# \\ D\end{matrix} \rightarrow \begin{matrix}F \\ a\end{matrix}$) the array is then sent to region 2 due to tin type and the hollow rectangle of $a's$ is formed by an application of the rule $D \rightarrow a$.

An incorrect application of the rule $\#C \rightarrow D\,a$ will end the generation with an application of the rule $\begin{matrix}\# \\ D\end{matrix} \rightarrow \begin{matrix}F \\ a\end{matrix}$, again sending the array to region 2 where it remains forever but not collected as a member in the language as it has a nonterminal F in it. It is known [5] that $R_h \in CD_2(REG, t)$. In fact the generation of a hollow rectangle of $a's$ in the $t-$communicating array P system given here is based on the technique used in [5] except that the type tin of the array P system is crucial in sending the array to region 2. This proves statement (1).

The statement (2) follows from (1) by noting that the set R_h of all hollow rectangles over a symbol a is not in ACF as no CF array grammar can generate it [26]. □

Theorem 3.4. 1. $S_h \in tEAP_5(REG) \cap CD_3(CFA, t)$

2. $S_h \notin CD_n(REGA, t), n \geq 1$

3. $tEAP_5(REG) - CD_n(REGA, t) \neq \emptyset, n \geq 1$

Proof:
The set S_h is generated by the following $t-$communicating array P system $tEAPS_5(REG)$

$$\Pi_4 = (\{U, R, U', R', X, Y, R'', U'', Z, F\}, \{a\}, [_1[_2[_3[_4[_5]_5]_4]_3]_2]_1, \left\{ \begin{matrix}U \\ X\,R\end{matrix} \right\},$$
$$\emptyset, \emptyset, \emptyset, \emptyset, R_1, R_2, R_3, R_4, R_5, 5)$$

$$R_1 = \{ \begin{matrix}\#\\U\end{matrix} \rightarrow \begin{matrix}U'\\a\end{matrix}\ ,\ R\# \rightarrow a\,R'\ \},$$
$$R_2 = \{\ U' \rightarrow U\ ,\ R' \rightarrow R\ ,\ Y \rightarrow X\ (out)\}$$
$$R_3 = \{\ X \rightarrow Y\ (out),\ X \rightarrow a\ \}$$
$$R_4 = \{\ U\#\# \rightarrow a\,a\,R''\ ,\ R''\# \rightarrow a\,R''\ ,\ R'' \rightarrow a\ ,\ \begin{matrix}\#\\\#\\R\end{matrix} \rightarrow \begin{matrix}U''\\a\\a\end{matrix}\ ,$$
$$\begin{matrix}\#\\U''\end{matrix} \rightarrow \begin{matrix}U''\\a\end{matrix}\ ,\ \begin{matrix}\#\\U''\end{matrix} \rightarrow \begin{matrix}Z\\a\end{matrix}\ ,\ \#Z \rightarrow F\,a\ \}$$
$$R_5 = \{\ Z \rightarrow a\ \}$$

Starting with the initial array $\left\{ \begin{matrix} U \\ X\,R \end{matrix} \right\}$ each of the leftmost column and the bottommost row of the hollow square of $a's$ is grown one step by the application of the rules $\begin{matrix}\#\\U\end{matrix} \rightarrow \begin{matrix}U'\\a\end{matrix}\ ,\ R\# \rightarrow a\,R'$ and the array is passed on to the region 2 due to the tin type of the system wherein the symbols U', R' are changed into U, R respectively and then sent to region 3, again due to the tin type of the system. If the rule applied is $X \rightarrow Y(out)$, then the array is immediately sent back to region 2 where the only rule applicable is $Y \rightarrow X(out)$ which sends it to region 1 and the process repeats.

But if the rule applied in region 3 is $X \rightarrow a$, then the array is sent to region 4 again due to the tin type of the system. In region 4, the uppermost side and rightmost side are grown. A correct application of the rules makes the growing uppermost side and the rightmost side meet at the right position thus yielding a hollow square of $a's$ except for the 'northeast' corner symbol which is Z. The tin type of the system sends the array to region 5 where the symbol Z is changed into a thus yielding a hollow square of $a's$ in the language generated. On the other hand an incorrect application of the rules in region 4 will send an array with a nonterminal F or U" to region 5 where it remains forever. This proves the statement (1). It is known [5] that $S_h \in CD_3(CFA, t)$. Statement (2) is due to [5]. Statement (3) follows from statements (1) and (2). □

Another model of grammars called Puzzle grammar has been introduced in [9] as an array generating two-dimensional grammar motivated by the problem of tiling the plane. It is known that context-free puzzle grammars (CFPG) and context-free array grammars (CFAG) coincide [9] whereas a subclass of CFPG, called Basic Puzzle grammars is known [21] to properly include the class of RAG's.

$$A \longrightarrow \overset{a}{\bigcirc}\ B\ ,\quad A \longrightarrow a\overset{B}{\bigcirc}\quad A \longrightarrow B\overset{a}{\bigcirc},\quad A \longrightarrow \overset{B}{\bigcirc}\ a$$

$$A \longrightarrow \begin{matrix}\overset{a}{\bigcirc}\\B\end{matrix}\ ,\ A \longrightarrow \begin{matrix}a\\\overset{B}{\bigcirc}\end{matrix}\ ,\ A \longrightarrow \begin{matrix}B\\\overset{a}{\bigcirc}\end{matrix}\ ,\ A \longrightarrow \begin{matrix}\overset{B}{\bigcirc}\\a\end{matrix}\ ,\ A \longrightarrow \overset{a}{\bigcirc}$$

A Basic Puzzle grammar consists of rules analogous to the rules of a regular array grammar but one of the symbols in the right side of the rule is circled. We give here the types of rules. (Here $A, B \in N$ and $a \in T$.)

Derivations begin with S written in a unit cell in the two-dimensional plane, with all the other cells containing the blank symbol $\#$, not in $N \cup T$. In a derivation step, denoted $\Rightarrow$, a non-terminal A in a cell is replaced by the right-hand member of a rule whose left-hand side is A. In this replacement, the circled symbol of the right-hand side of the rule used, occupies the cell of the replaced symbol and the non-circled symbol of the right side occupies the cell to the right or the left or above or below the cell of the replaced symbol depending on the type of rule used. The replacement is possible only if the cell to be filled in by the non-circled symbol contains a blank symbol.

The set of pictures or arrays generated by G, denoted $L(G)$, is the set of connected, digitized finite arrays over T, derivable in one or more steps from the axiom.

In [22] cooperating basic puzzle grammar systems with components having basic puzzle grammar rules and maximal mode of derivation has been considered and their picture generating power has been examined. The family of array languages generated by such systems with at most n components in the maximal derivation mode is denoted by $CD_n(BPG, t)$. On the other hand in [23] array P systems with array objects and basic puzzle grammar rules in the regions of the system are considered. These P systems are called as BPG array P systems. Here we can incorporate $t-$communication in these BPG array P systems. We denote by $tEAPS_m(BPG)$ such an array P system having at most m membranes and having BPG rules and array objects in its regions and type tin. The family of array languages generated by such systems is denoted by $tEAP_m(BPG)$

Let R_{hf} be the set of all hollow rectangular frames over the symbol $\{a\}$, a member of which is shown in Figure 4.

$$\begin{array}{cccccccc} & a & & & & & a & \\ a & a & a & a & a & a & a & a \\ & a & & & & & a & \\ & a & & & & & a & \\ a & a & a & a & x & a & a & a \\ & a & & & & & a & \end{array}$$

Figure 4 : A Rectangular frame of $a's$

Theorem 3.5. 1. $R_{hf} \in tEAP_2(BPG) \cap CD_2(BPG, t)$

2. $tEAP_2(BPG) - ACF \neq \emptyset$

Proof:
The set R_{hf} is generated by the following $t-$communicating array P system $tEAPS_2(BPG)$

$\Pi_5 = (\{S, A, B, C, D, E, I, J, K, X, Y, Z\}, \{a\}, [_1\, [_2]_2]_1, \{S\}, \emptyset, \emptyset, R_1, R_2, 2)$

$$P_1 = \{S \to \overset{S}{(a)}, \quad S \to \overset{A}{(a)}, \quad A \to a\,(B)\ , \quad B \to \overset{a}{(C)}\ ,$$

$$C \to (a)C\ , \quad C \to (D)\,a\ , \quad D \to \overset{a}{(E)}, \quad E \to \underset{F}{(a)}\ ,$$

$$I \to \overset{Z}{(J)}a\ , \quad J \to J\,(a), \quad J \to K\,(a), \quad K \to (a),$$

$$F \to \underset{F}{(a)}\ , \quad F \to \underset{a}{(I)}\}$$

$$P_2 = \{K \to \underset{a}{(X)}\ , \quad X \to a\,(Y)\ , \quad Y \to (a)\}$$

It can be seen that R_{hf} is generated by the $t-$communicating array P system Π_5. The generation will proceed analogous to the generation of a rectangle in the proof of Theorem 3.4 except that the 'protrusions' in the corners will be produced. It is known [23] that $R_{hf} \in CD_2(REG, t)$. In fact the generation of a hollow rectangular frame of $a's$ in the $t-$communicating array P system given here is based on the technique used in [23] except that the type tin of the array P system sends the array to an inner region. This proves statement (1).

The statement (2) follows from (1) by noting that the set R_{hf} of all hollow rectangular frames over a symbol a is not in ACF as no CF array grammar can generate it [5]. □

4. Conclusion

We have examined in this paper the relationship between the two array generating models, namely array grammar systems and array P systems by extending the $t-$derivation mode of cooperating array grammar systems [5] to rewriting P systems [1] with array objects and array-rewriting rules. We have also compared the power of the $t-$communicating array P systems with other array grammar models. It remains to be seen whether the number of membranes used in the examples and results could be reduced. Although the tin type of the system considered here has proved to be very useful in view of the halting condition in computations, it remains to examine the power of the other type $tout$. This might require a different approach in the way of defining successful computations.

Acknowledgements

The authors would like to thank the referees for their very useful comments which improved the presentation of the paper. The authors are especially grateful to one of the referees for pointing out a correction

in Theorem 3.2. The first two authors K.G. Subramanian and Rosihan M. Ali acknowledge support from FRGS grant and RU grant of the Universiti Sains Malaysia for this research.

References

[1] R. Ceterchi, M. Mutyam, Gh. Păun and K.G. Subramanian, Array - rewriting P systems, *Natural Computing*, **2** (2003), 229-249.

[2] C.R. Cook, P.S.-P. Wang , A Chomsky hierarchy of isotonic array grammars and languages, *Computer Graphics and Image Processing*, **8** (1978), 144-152.

[3] E. Csuhaj-Varju, J. Dassow, J.Kelemen, Gh. Păun, *Grammar systems: A grammatical approach to distribution and cooperation* , Gordon and Breach Science Publishers, 1994.

[4] E. Csuhaj-Varju, G. Vaszil, Gh. Păun, Grammar systems versus membrane computing: The case of CD grammar systems, *Fundamenta Informaticae*, **76** (2007), 271-292.

[5] J. Dassow, R. Freund, Gh. Păun , Cooperating array grammar systems,*Int. J. of Pattern Recognition and Artificial Intelligence*, **9** (1995), 1029-1053.

[6] A. Ehrenfeucht, Gh. Păun, G. Rozenberg, Contextual grammars and formal languages. In: G. Rozenberg and A. Salomaa (eds.), , *Handbook of Formal Languages*, Springer-Verlag, Vol. **2** (1997), 237-293.

[7] D. Giammarresi, A. Restivo, Two-dimensional languages, In: G. Rozenberg and A. Salomaa (eds.), , *Handbook of Formal Languages*, Springer Verlag, Vol.**3**, (1997), 215 - 267.

[8] K. Krithivasan, Variations of the matrix models, *Int. J. Computer Mathematics*, **6** (1977), 171-190.

[9] M. Nivat, A. Saoudi, K.G. Subramanian, R. Siromoney, R. and V.R. Dare, Puzzle grammars and context-free array grammars, *Int. J. Pattern Recognition and Artificial Intelligence*, **5** (1991), 663-676.

[10] Gh. Păun, *Marcus Contextual Grammars*, Kluwer Academic Publishers, Dordrecht-Boston-London, 1997.

[11] Gh. Păun, *Membrane Computing : An Introduction*, Springer-Verlag Berlin, Heidelbrg, 2000.

[12] S.C. Reghizzi and M. Pradella, Tile rewriting grammars and picture languages, *Theoretical Computer Science*, **340** (2005), 257-272.

[13] A. Rosenfeld, *Picture Languages*, Academic Press, Reading, MA, 1979.

[14] A. Rosenfeld, R. Siromoney, Picture Languages-A Survey, *Languages of Design*, **1** (1993), 229-245.

[15] G. Rozenberg and A. Salomaa, *The Mathematical Theory of L systems*, Academic Press, New York, 1980.

[16] A. Salomaa, *Formal Languages*, Academic Press, New York, 1973.

[17] G. Siromoney, R. Siromoney, K. Krithivasan, Abstract families of matrices and picture languages, *Computer Graphics and Image Processing*, **1** (1972), 234-307.

[18] R. Siromoney, K.G. Subramanian, K. Rangarajan, Parallel/Sequential rectangular arrays with tables, *Int.J. Computer Mathematics*, **6** (1977) 143-158.

[19] R. Stiebe, Slender Siromoney Matrix Languages, *Proceedings of the* 1^{st} *International Conference on Language and Automata: Theory and Applications*, Tarragona, Spain, 2007.

[20] K.G. Subramanian, R. Siromoney, G. Siromoney, A Note on an Extension of Matrix Grammars Generating Two-dimensional Languages, *Information Sciences*, **35** (1985), 223-233.

[21] K.G. Subramanian, R. Siromoney, V.R. Dare, and A. Saoudi, Basic Puzzle Languages, *Int.J. Pattern Recognition and Artificial Intelligence*, **9** (1995), 763-775.

[22] K.G. Subramanian, R. Saravanan, P. Helen Chandra, Cooperating Basic Puzzle Grammars, *Lecture Notes in Computer Science*, **4040** (2006), 354-360.

[23] K.G. Subramanian, R. Saravanan, M. Geethalakshmi, P. Helen Chandra, M. Margenstern, P systems with array objects and array rewriting rules, *Progress in Natural science*, **17** (2007) 479-485.

[24] K.G. Subramanian, D. L. Van, P. Helen Chandra, N. D. Quyen, Array Grammars with Contextual Operations, *Fundamenta Informaticae*, **83** (2008), 411-428.

[25] P.S.P. Wang, Some new results on isotonic array grammars, *Information Processing Letters*, **10** (1980), 129-131.

[26] Y. Yamamoto, K. Morita, K. Sugata, Context-sensitivity of two-dimensional regular array grammars, In P.S.-P. Wang, ed., *Array Grammars, Patterns and Recognizers*, WSP Series in Computer Science, **18**, World Scientific Publ., Singapore (1989), 17-41.